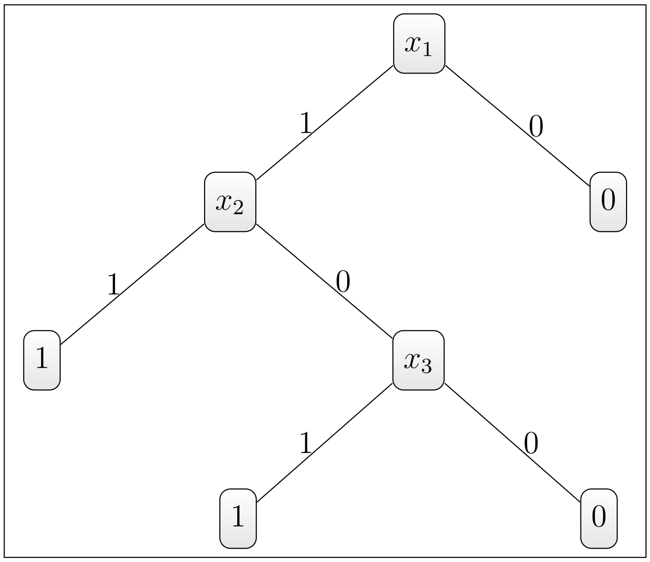
CS 5350/6350: Machine Learining Fall 2017

Homework 1 Solution
UNID: u1143683

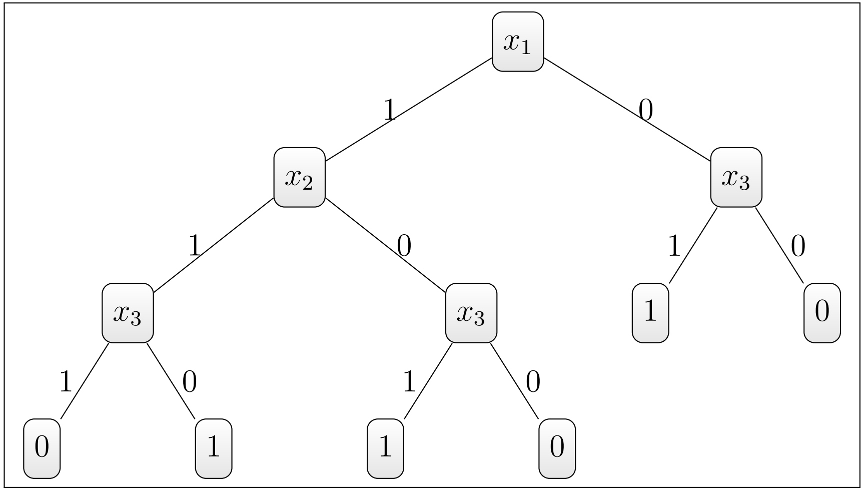
September 11, 2017

1 Decision Tree

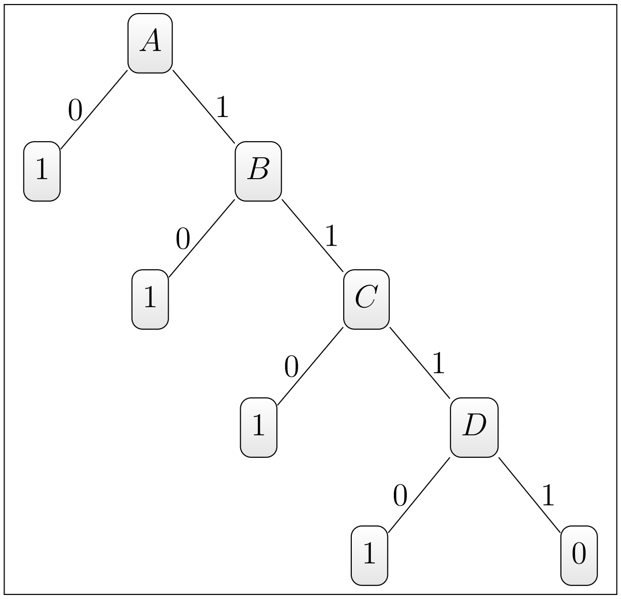
1. (a) $(x_1 \wedge x_2) \vee (x_1 \wedge x_3)$



(b) $(x_1 \vee x_2)$ xor x_3



(c) $\neg A \vee \neg B \vee \neg C \vee \neg D$



2. (a)

(b) $S = 9$
$p_+ = 5/9 = 0.56$
$p_- = 4/9 = 0.44$

$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$
$Entropy(S) = -0.56 \log_2 0.56 - 0.44 \log_2 0.44$
$Entropy(S) = 0.99$

(c) Information gain of an attribute A is given by,

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

i. Feature $A =$ Technology

For $v = Yes$,
$S_v = 3$
$p_+ = 1/3 \quad p_- = 2/3$
$Entropy(S_v) = 0.918$

For $v = No$,
$S - v = 6$
$Entropy(S_v) = 0.918$

$Gain(S, Technology) = 0.99 - (6/9)0.918 - (3/9)0.918 = 0.071$

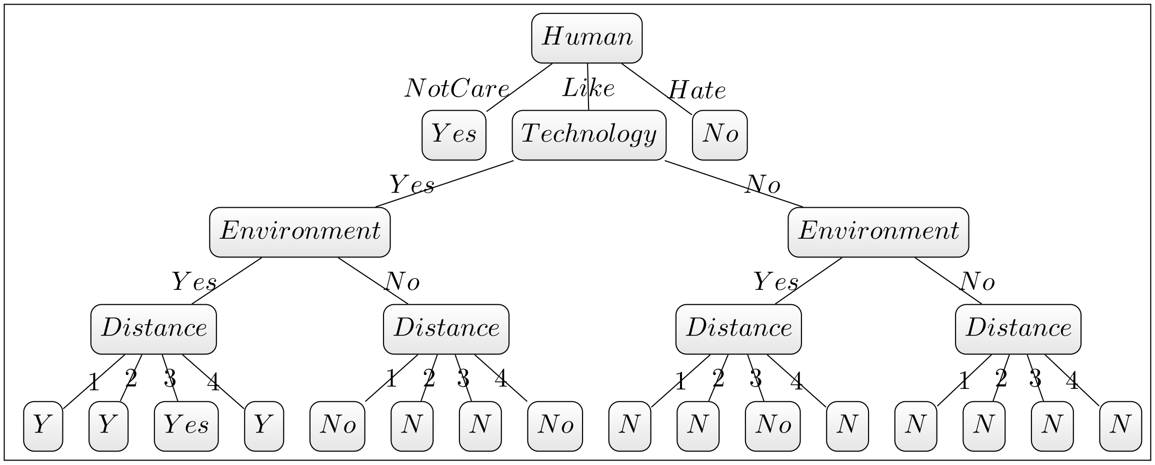
ii. Feature $A =$ Environment
$Gain(S, Environment) = 0.228$

iii. Feature $A =$ Human
$Gain(S, Human) = 0.629$

iv. Feature $A = Distance$
$Gain(S, Distance) = 0.155$

(d) The attribute $Human$ provides the best prediction of target attribute $Invade$ according to the information gain measure. So I will use attribute $Human$ as root node.

(e) Decision Tree using Human Attribute as root node.



(f) Predictions:

Technology	Environment	Human	Distance	Invade(Prediction)	Invade (Given)
Yes	Yes	Like	2	Yes	No
No	No	Hate	3	No	No
Yes	Yes	Lkie	4	Yes	Yes

Accuracy is 2/3

2 Linear Classifier

1. The given table can be represented by the following linear classifier:

$$x_1 + x_4 \geq 1$$

2. Classification:

x1	x2	x3	x4	o(prediction)	o(given)
0	0	0	1	1	1
0	0	1	1	1	1
0	0	0	0	-1	-1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1
1	1	1	0	1	1

Classifier accuracy = 7/7

3. The combined dataset:

x_1	x_2	x_3	x_4	o
1	0	1	1	1
0	1	0	1	1
0	0	1	0	-1
0	0	0	1	1
0	0	1	1	1
0	0	0	0	-1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1
1	1	1	0	1
0	1	0	0	-1
0	1	1	0	-1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	1	0	1	1

The classifer,

$$x_1 + x_4 \geq 1$$

fits the above truth table perfectly.

3 Experiments

In this question you will be implementing a decision tree learner. You will experiment with the decision tree hyperparameters using cross-validation.

There is a secret computer science conference that only a selected group of computer scientists will be invited. Fortunately, we have a secret agent that has access to the guest list. But, she only has part of it. The accessible name list is in the **Dataset/training.data** file. Your job is to use this training data to learn a decision tree classifier that can predict if the names in the **Dataset/test.data** file will be invited to the secret conference. We suggest some features for the dataset, but you need to extract the features yourself. You are also welcome to add your own features.

You may use any programming language for your implementation. However, the graders should be able to execute your code on the CADE machines.

Cross-Validation

The depth of the tree is a hyper-parameter to the decision tree algorithm that helps reduce overfitting. You will see later in the semester that many machine learning algorithm (SVM, logistic-regression etc) have some hyper-parameters as their input. One way to determine a proper value for the hyper-parameter is to use a technique called cross-validation.

As usual we have a training set and a test set. Our goal is to discover good hyper-parameters using the training set. To do so, you can put aside some of the training data aside, and when training is finished, you can test the resulting classifier on the held out data. This allows you to get an idea of how well the particular choice of hyper-parameters does. However, since you did not train on your whole dataset you may have introduced a statistical bias in the classifier. To correct for this, you will need to train many classifiers with different subsets of the training data removed and average out the accuracy across these trials.

For problems with small data sets, a popular method is the leave-one-out approach. For each example, a classifier is trained on the rest of the data and the chosen example is then evaluated. The performance of the classifier is the average accuracy on all the examples. The downside to this method is for a data set with n examples you must train n different classifiers. Of course, this is not practical for the data set you will use in this problem, so you will hold out subsets of the data many times instead.

Specifically, for this problem, you should implement k-fold cross validation. The general approach for k-fold cross validation is the following: Suppose you want to evaluate how good a particular hyper-parameter is. You split the training data into k parts. Now, you will train your model on k 1 parts with the chosen hyper-parameter and evaluate the trained model on the remaining part. You should repeat this k times, choosing a different part for evaluation each time. This will give you k values of accuracy. Their average cross-validation accuracy gives you an idea of how good this choice of the hyper-parameter is. To find the best value of the hyper-parameter, you will need to repeat this procedure for different choices of the hyper-parameter. Once you find the best value of the hyper-parameter, use the value to retrain you classifier using the entire training set.

1. [25 points] **Implementation**

For this problem, your will be using the data in **Dataset** folder. This folder contains two files: **training.data** and **test.data**. You will train your algorithm on the training file. Remember that you should not look at or use your testing file until your training is complete.

(a) [8 points] Implement the decision tree data structure and the ID3 algorithm for your decision tree (Remember that the decision tree need not be a binary tree!). For debugging your implementation, you can use the previous toy examples like the alien data from Table **??**. Discuss what approaches or choices you had to make during this implementation.

(b) [4 points] Suggest at least 4 other features you could have extracted from this dataset.

(c) [2 points] Report the error of your decision tree on the **Dataset/training.data** file.

(d) [5 points] Report the error of your decision tree on the **Dataset/test.data** file.

(e) [1 points] Report the maximum depth of your decision tree.

2. [20 points] **Limiting Depth**

   In this section, you will be using 4-fold cross-validation in order to limit the depth of your decision tree, effectively pruning the tree to avoid overfitting. You will be using the 4 cross-validation files for this section, titled Dataset/CVSplits/training 0X.data where X is a number between 0 and 4 (inclusive)

   (a) [10 points] Run 4-fold cross-validation using the specified files. Experiment with depths in the set $\{1, 2, 3, 4, 5, 10, 15, 20\}$, reporting the cross-validation accuracy and standard deviation for each depth. Explicity specify which depth should be chosen as the best, and explain why.

   (b) [5 points] Using the depth with the greatest cross-validation accuracy from your experiments: train your decision tree on the **Dataset/training.data** file. Report the accuracy of your decision tree on the **Dataset/test.data** file.

   (c) [5 points] Discuss the performance of the depth limited tree as compared to the full decision tree. Do you think limiting depth is a good idea? Why?

Experiment Submission Guidelines

1. The report should detail your experiments. For each step, explain in no more than a paragraph or so how your implementation works. You may provide the results for the final step as a table or a graph.

2. *Your code should run on the CADE machines.* You should include a shell script, `run.sh`, that will execute your code in the CADE environment. Your code should produce similar output to what you include in your report.

   You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.

3. Please do not hand in binary files! We will *not* grade binary submissions.

4 Decision Lists (For CS 6350 students)

Theorem. *Any 1-decision list is a linearly separable function.*

Proof. We will prove the above statement by induction on terms in decision list. Let us assume that any 1-decision list is a linearly seperable function.
Let's consider the base case with only one term x_1. For now we assume x_1 is non negated. We have four different decision lists possible for this scenario. We will select a weight vector $\boldsymbol{w}$ and bias b for each of the case, hence representing the decision list as a linearly separable function.

1. $(x_1, 1), (T, 1)$
   $\boldsymbol{w} = [2]$
   $b = 1$
   We get,
   $$2x_1 + 1 > 0$$
2. $(x_1, 1), (T, 0)$
   $\boldsymbol{w} = [2]$
   $b = -1$
   We get,
   $$2x_1 - 1 > 0$$
3. $(x_1, 0), (T, 1)$
   $\boldsymbol{w} = [-2]$
   $b = 1$
   We get,
   $$-2x_1 + 1 > 0$$
4. $(x_1, 0), (T, 0)$
   $\boldsymbol{w} = [-2]$
   $b = -1$
   We get,
   $$-2x_1 - 1 > 0$$

In case x_1 is negated, we can replace x_1 in the equation with $(1 - x_1)$. So our assumption is true for our base case with one term in decision tree. Now lets assume that any 1-decision list with n terms is also a linearly separable function. Suppose we have a 1-decision list with $n + 1$ terms,

$$f = (a_1, c_1), (a_2, c_2), (a_3, c_3), \ldots.., (a_{n+1}, c_{n+1})$$

By induction hypothesis the decision list

$$(a_2, c_2), (a_3, c_3), \ldots.., (a_{n+1}, c_{n+1})$$

of length n is a linearly separable function. Let it be represeted by weight vector $\boldsymbol{w}$ and bias b. Let $||w||_1 = \sum_{i=1}^{n} |w_i|$ be the 1-norm of $\boldsymbol{w}$. There are now four different values the first term (a_1, c_1) can take as follows: $(x_1, 1), (x_1, 0), (\bar{x_1}, 1), (\bar{x_1}, 0)$.

Let $e_1 = [1, 0, 0, 0....0]$ be a vector and $A = ||w|| + |b| + 1$. Now we claim that decision list f is a linearly separable function represented by weight vector w' and bias b' where,

$$w' = w + Ae_1, \ b' = b$$

$$w' = w - Ae_1, \ b' = b$$

$$w' = w - Ae_1, \ b' = b - A$$

$$w' = w + Ae_1, \ b' = b - A$$

To verify our claim let us consider case where (a_1, c_1) takes the form $(x_1, 1)$. For $x \in \{0, 1\}^n$.

$$\langle w', x\rangle = \langle w + Ae_1, x\rangle = \langle w, x\rangle + Ax_1$$

Now,

$$\langle w', x\rangle \geq b' = b$$

$$\langle w, x\rangle + Ax_1 \geq b$$

If $x_1 = 1$ we have, $\langle w, x\rangle - A \geq b$

$$\langle w, x\rangle + ||w|| + |b| + 1 \geq b$$

$$\langle w, x\rangle \geq -||w|| - 1$$

Now, $||w|| + 1 > ||w||$

$$\langle w, x\rangle > -||w||$$

Above statement is true for $x \in \{1, 0\}^n$, so output of the function is 1. Now consider $x_1 = 0$, we have

$$\langle w, x\rangle \geq b$$

By inductive hypothesis the decision list,

$$f' = (a_2, c_2), (a_3, c_3), ..., (a_{n+1}, c_{n+1})$$

is linearly separable function with weight vector $\boldsymbol{w}$ and bias b. Hence output of the linear separable equation is 1 only when output of decision list f' is 1, which is how f calculates its output when $x_1 = 0$. we can say f is indeed a linearly separable function. Other claims can be verified in a similar manners.

□